Mandatory assignment 2 - TEK4030

Exercise 1 - Force Control - Simulink

In this exercise we will use different force control methods to control the planar arm already modelled in the Motion Control exercise in assignment 1. Only the manipulator model can be reused, as the controllers are joint space controllers when the force control schemes use operational space control laws. Before we can implement the force control methods, the model needs to be expanded with a model of the environment. We will model the environment as a frictionless elastic plane, as in Example 9.1 in the textbook.

We assume that the end effector can move freely as long as the end effector is located on the left side of the plane, or on other words $x_e < x_r$. Here x_e is the end effector position along the x-axis and x_r is the environment rest position. The environment model must take this into account. The environment can be modeled as

$$f_{e,x}(x_e) = \begin{cases} 0 & x_e < x_r \\ k_x(x_e - x_r) & x_e \geq x_r \end{cases} \tag{1}$$

a) Expand the manipulator model in Simulink by including the model of the environment (i.e. implement (9.1) in the textbook). In order to calculate the force from the enironment you will need the end effector position, therefore you must also find the forward kinematics for the robot. Chapter 2.9.1 in the textbook will give the forward kinematics of a three-link planar arm robot. By setting $\theta_3 = 0$ and $a_3 = 0$ you will get the forward kinematics of a two-link planar arm robot.

b) Before implementing any controllers we will need the analytic Jacobian for the two-link planar arm. In Section 3.2.1 you will find the geometric Jacobian of a three-link planar arm robot. By setting $a_3 = 0$ you will get the geometric Jacobian of a two-link planar arm robot. Use the operational space state

$$\boldsymbol{x}_e = \begin{bmatrix} x_e \\ y_e \\ \phi \end{bmatrix} \tag{2}$$

where x and y represents the Cartesian position and ϕ the orientation ($\phi = 0$ equals pointing along the x-axis) of the end-effector. ϕ represents rotation around the z-axis, thus $\dot{\phi} = \omega_z$. Find the analytic Jacobian for the two-link planar arm, i.e. $\boldsymbol{J}_A$ in the expression $\dot{\boldsymbol{x}}_e = \boldsymbol{J}_A \dot{\boldsymbol{q}}$. The matrix should be a 3x2 matrix. For the manipulator in this exercise it can be shown that the geometric and analytic Jacobian are equal, assuming we define $\boldsymbol{v}_e = [v_x \; v_y \; \omega_z]^T$ (see (3.39) in the textbook).

c) Simulate the active compliance control law, i.e. (9.15) in the textbook, in Simulink. Since we only have to degrees of freedom we reduce the operational space state to only two states

$$\boldsymbol{x}_e = \begin{bmatrix} x_e \\ y_e \end{bmatrix} \tag{3}$$

By removing ϕ we will have to remove the last row in the analytic Jacobian found in b). Due to the simple geometry of the problem, involving only position variables, all the quantities can be referred to the base frame. Use $x_r = 0.5$ and set $q(t = 0) = [0.75\pi, -0.75\pi]^T$, $\boldsymbol{x}_d = [0.75, 0.25]^T$ and $k_x = 100$. Simulate the system with different values of $\boldsymbol{K}_P$. What are the effects on the steady-state end-effector position when increasing/decreasing $\boldsymbol{K}_P$? Report plots for x_e for two different $\boldsymbol{K}_P$ to support your argument. (Tip: have a look at example 9.1)

d) In order to implement impedance control we will need the time derivative of the analytic Jacobian. Find the the time derivative of the analytic Jacobian ($\dot{\boldsymbol{J}}_A$), based on the 2x2 analytic Jacobian used in c).

e) Simulate the impedance control law in (9.30) and (9.31) in Simulink. Use the operational space state $\boldsymbol{x}_e = [x_e, y_e]^T$, as before. Due to the simple geometry of the problem, involving only position variables, all the quantities can be referred to the base frame. Use $\boldsymbol{M}_d$, $\boldsymbol{K}_D$ and $\boldsymbol{K}_P$ as given in Example 9.2, and use the initial conditions from c). Use the same parameters as in c), but use a trajectory instead. Let $\ddot{\boldsymbol{x}}_d = [0.1, 0.05]^T$ for the time step $t = 0$ to $t = 1$, and $\ddot{\boldsymbol{x}}_d = [-0.1, -0.05]^T$ for the time step $t = 5$ to $t = 6$. Report plots of $\boldsymbol{x}_e$.

f) Try varying $\boldsymbol{M}_d$, $\boldsymbol{K}_D$ and $\boldsymbol{K}_P$ individually. Report the effect of increasing/decreasing the parameters.

g) Simulate the force control with inner velocity loop control law (9.45). Use the same parameters where appropriate. Set the desired force setpoint to $\boldsymbol{f}_d = [5, 0]^T$. Find control parameters $\boldsymbol{K}_F$ that you think are good for $k_x = 100$. Report the parameters and plots of the force. What will happen with the performance of the controller if k_x changes?

Exercise 2 - Visual servoing - ROS

In this exercise you will implement *Image-based Visual Servoing* using the *Resolved-velocity control* scheme. A simulator has been implemented using Gazebo in ROS. Have a look at the webpage below for how to download the simulator
`https://github.uio.no/TEK4030/tek4030_ros_intro/blob/master/oblig_2.md` In this exercise you should implement a controller node for a camera. The camera can move arbitrary in space in the simulator, but in a real scenario it would be attached to a robot.

a) Implement a ROS node subscribing to the following topics

/imgproc/points_normalized This is the normalized image points. Each point is denoted X and Y in the textbook, and the whole vector of $X\ Y]$ is denoted $\boldsymbol{s}$, which is the feature vector.
Type `tek4030_visual_servoing_msgs::ImageFeaturePoints`

The node should publish the following topics. For now you should just make the code for publishing these types, and the calculation of the values will come later

camera_twist The desired movement of the camera according to the control law, $\boldsymbol{v}_r^c$.
Type `geometry_msgs::Twist`

/imgproc/points_setpoint This is the normalized image set point, denoted $\boldsymbol{s}_d$ in the textbook.
Type `tek4030_visual_servoing_msgs::ImageFeaturePoints`

/imgproc/points_error This is the feature vector error, denoted $\boldsymbol{e}_s$ in the textbook.
Type `tek4030_visual_servoing_msgs::ImageFeaturePoints`

b) Implement the control law (10.89) in the textbook. Use the set point $\boldsymbol{s}_d = [0.15, 0.15, -0.15, 0.15, -0.15, -0.15, 0.15, -0.15]^T$ and use $z_c = 1.0$ for all points when calculating the Interaction matrix L_s. Publish the desired velocity, the set point and the error. Simulate the system and plot all the errors in rqt_plot.

c) Simulate the system again using the set point $\boldsymbol{s}_d = [0.3, -0.3, 0.3, 0.3, -0.3, 0.3, -0.3, -0.3]^T$. Plot all the errors in rqt_plot.

Exercise 3 - Visual servoing

a) Explain the principal difference between image-based and position-based visual servoing.

b) What is the benifit of using resolved velocity control when doing visual servoing?

c) What does the image Jacobian and the interaction matrix represent?

d) Prove the stability of the image-based visual servoing control law (10.84). This means showing the steps between (10.80), (10.81) and (10.85).

Exercise 4 - Tele-operations

a) Explain the difference between unilateral and bilateral tele-operation.

b) What does the hybrid matrix represent, and what is a transparent tele-operation system like?

c) Briefly explain the forward-flow tele-operation controller. Is it unilateral or bilateral?

Submission

Submit a small report answering the exercises and all the questions in the exercises. Also attach your all your code. Submit your assignment in Canvas (`canvas.uio.no`), bu uploading your report as a separate pdf-file, and your source code as a separate zip-file.